

Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

1. (Currently amended) A processor, the processor implemented as a three way superscalar, pipelined architecture, the processor comprising:
an out-of-order microinstruction pointer (μ IP) stack for storing pointers in a microcode (μ code) execution core, a plurality of the pointers associated with a common macroinstruction, the plurality of pointers placed on the out-of order microinstruction pointer stack and removed from the microinstruction pointer stack before it is known if a sequence of microinstructions pointed to by the plurality of pointers is valid.

2. (Previously presented) The processor of claim 1 in which entries in the μ IP stack comprise:

an entry number field;
a microinstruction pointer (μ IP) field;
a back pointer field;
a retirement indicator field; and
a return pointer field.

3. (Original) The processor of claim 2 in which the μ IP field is 14-bits wide.

4. (Original) The processor of claim 3 in which the μ IP field has a microinstruction pointer (μ IP) pushed by a first microoperation (μ Op) code and used by a second μ Op code.

5. (Original) The processor of claim 2 in which the back pointer field has a pointer to a next entry in the μ IP stack for a micro-type of service (μ TOS) bit to point to after a μ Op.

6. (Original) The processor of claim 2 in which the retirement indicator field has an indication of whether an entry has retired.

7. (Original) The processor of claim 2 in the return pointer field a pointer to a location in a retirement stack to which an entry is copied after being retired.

8. (Currently amended) A method executed in a processor, the processor implemented as a three way superscalar, pipelined architecture, the method comprising:

executing microcode (μ code) addressed by pointers stored in an out-of-order microinstruction pointer (μ IP) stack, a plurality of pointers associated with a common macroinstruction, the plurality of pointers placed on the out-of-order microinstruction pointer stack and removed from the microinstruction pointer stack before it is known if a sequence of microinstructions pointed to by the plurality of pointers is valid; and manipulating the μ IP stack with a set of microinstructions.

9. (Previously presented) The method of claim 8 in which entries in the stack have an entry number field, a microinstruction pointer (μ IP) field, a back pointer field, a retirement indicator field and a return pointer field.

10. (Original) The method of claim 9 in which the μ IP pointer field is 14-bits wide.

11. (Original) The method of claim 10 in which the μ IP pointer field has a microinstruction pointer (μ IP) pushed by a first microoperation (μ Op) code and used by a second μ Op code.

12. (Original) The method of claim 9 in which the back pointer field has a pointer to a next entry in the μ IP stack for a micro-type of service (μ TOS) bit to point to after a μ Op.

13. (Original) The method of claim 9 in which the retirement indicator field has an indication of whether an entry has retired.

14. (Original) The method of claim 9 in which the return pointer field contains a pointer to a location in a retirement stack to which an entry is copied after being retired.

15. (Original) The method of claim 9 in which manipulating comprises:
pushing a next μ IP on to the μ IP stack; and
using the next μ IP in an intermediate field as a target μ IP in a jump operation.

16. (Original) The method of claim 9 in which manipulating comprises:
taking a value of an intermediate field of a microoperation (μ Op); and
pushing the value on to the μ IP stack.

17. (Original) The method of claim 9 in which manipulating comprises:
popping a value off the μ IP stack; and
replacing a current μ Op intermediate field.

18. (Original) The method of claim 9 in which manipulating comprises:
popping a value off of the μ IP stack; and
jumping to that value.

19. (Original) The method of claim 9 in which manipulating comprises:
reading a value off the μ IP stack; and
replacing a μ Op's intermediate field with the value.

20. (Original) The method of claim 9 in which manipulating comprises setting the μ IP stack pointers to reset.

21. (Original) The method of claim 9 further comprising providing a set of pointers that point to different entries in the μ IP stack.

22. (Original) The method of claim 21 in which the set of pointers includes a μ TOS pointer that points to a top of the μ IP stack.

23. (Original) The method of claim 21 in which the set of pointers includes a μ Alloc pointer that points to a next allocated entry in the μ IP stack.

24. (Original) The method of claim 21 in which the set of pointers includes a NextRet pointer that points to a next entry in the μ IP stack to be deallocated.

25. (Original) The method of claim 21 in which the set of pointers includes μ RetTos pointer that points at a retired top of the μ IP stack.

26. (Original) The method of claim 8 in which the μ OPs include an ms_call μ OP that takes a next μ IP, pushes the next μ IP on the μ IP stack, and uses the next μ IP in an intermediate field as a target μ IP of a jump.

27. (Original) The method of claim 8 in which the μ OPs include an ms_push μ OP that takes a value in an intermediate field and pushes the value on the μ IP stack.

28. (Original) The method of claim 8 in which the μ OPs include an ms_pop μ OP that pops a value off the μ IP stack and replaces the value with the μ OP's intermediate field.

29. (Original) The method of claim 8 in which the μ OPs include an `ms_return` μ OP that pops a value off of the μ IP stack and jumps to that μ IP.

30. (Original) The method of claim 8 in which the μ OPs include an `ms_tos_read` μ OP that reads a value off the μ IP stack and replaces this μ OP's intermediate field.

31. (Original) The method of claim 8 in which the μ OPs include an `ms_ μ ip_stack_clear` μ OP that sets the μ IP stack pointers to reset.

32. (Currently amended) A computer program product residing on a computer readable medium having instructions stored thereon which, when executed by the processor, cause the processor to:

execute microcode (μ code) addressed through pointers stored in an out-of-order microinstruction pointer (μ IP) stack, a plurality of the pointers associated with a common macroinstruction, the plurality of pointers placed on the out-of-order microinstruction pointer stack and removed from the microinstruction pointer stack before it is known if a sequence of microinstructions pointed to by the plurality of pointers is valid; and
manipulate the μ IP stack with a set of microinstructions.

33. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

push a next μ IP on to the μ IP stack; and
use the next μ IP in an intermediate field as a target μ IP in a jump operation.

34. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

take a value of an intermediate field of a microoperation (μ Op); and

push the value on to the μ IP stack.

35. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

pop a value off the μ IP stack; and
replace a current μ Op intermediate field with the value.

36. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

pop a value off of the μ IP stack; and
jump to that value.

37. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

read a value off the μ IP stack; and
replace a μ Op's intermediate field with the value.

38. (Original) The computer program product of claim 32 wherein instructions to manipulate further comprise instructions to:

set the μ IP stack pointers to reset.